Software Requirements Specification (SRS)

Project: Photo Atlas — Visual Memory of the City

---

1. Introduction

1.1 Purpose

The purpose of the **Photo Atlas** system is to create a structured, searchable digital archive of historical photographs that capture the visual memory of the city, particularly Istanbul. The system will automate photo tagging, enable spatial visualization on a map, and support both public and private access to images. It will serve as a cultural encyclopedia of urban imagery for researchers, historians, and citizens.

1.2 Scope

The platform will:

- Store and organize historical photographs with metadata such as **location, time, and object identification**.
- Provide **automated tagging and similarity detection** for newly uploaded photos.
- Allow **privacy-controlled uploads** for personal and family photos.
- Integrate with **existing collections** (e.g., Eski İstanbul) and facilitate **visual and spatial exploration**.
- Enable **interactive annotations**, **user suggestions**, and **mobile-based location experiences**.

The system will be accessible via web and mobile interfaces, using a cloud-based infrastructure.

1.3 Definitions and Abbreviations

Term	Definition
FOV	Field of View
Tagging	Assigning descriptive metadata (location, building, year, etc.)
Private Photo	User-restricted access to personal or family photos
Photo Atlas	Main web platform for visual and spatial organization
Culture Atlas ID	Unique identifier linking photo locations to cultural sites

---

2. Current System Overview

The current workflow is mostly **manual**, with users uploading and tagging photos directly into WordPress.

- Location and FOV data are manually derived by observing the image.
- The oldest photos date back to **1843**, with more recent ones from the 1990s.
- **13900 photos** are currently uploaded.
- There is **no distinction between public/private photos**.
- Users manually infer time periods based on contextual clues like **car models, clothing, and architecture**.
- Photos are primarily stored in **cloud-based WordPress storage**.

---

3. System Objectives

1. **Automation:** Reduce manual labor through automated tagging and similarity detection.
2. **Accessibility:** Create a publicly accessible, searchable photo encyclopedia.
3. **Interactivity:** Enable map-based exploration, tagging, and user participation.
4. **Privacy & Security:** Support private photo uploads and access control.
5. **Scalability:** Allow integration with existing archives and future datasets.

---

4. Functional Requirements

4.1 Photo Upload and Management

- Users can upload both **public** and **private** photos.
- Metadata fields include **title, description, location, FOV, date range, and license info**.
- When uploading, the system displays **visually similar existing photos** to assist with identification.

4.2 Automatic Tagging and Recognition

- AI-powered tagging for **location, date estimation**, and **object/building identification**.
- Tags are stored as searchable metadata.
- Unknown photos are suggested for verification via similarity to existing ones.

4.3 Map and Spatial Interface

- Each photo is geolocated and displayed on a map interface.
- Map tags reference **Culture Atlas IDs** for known landmarks.
- Photo markers show thumbnail previews with tag visibility toggles (visible/invisible).
- Clicking a landmark tag displays all related photos.

4.4 User Roles and Permissions

- **Admin:** Approves or rejects community suggestions; manages tags and users.
- **Contributor:** Uploads and tags photos.
- **Viewer:** Explores public photos and submits suggestions.
- Private photos are viewable only by their owners and authorized accounts.

4.5 Suggestion System

- Users can suggest corrections or add missing information (e.g., location, date).
- Suggestions are queued for **admin review**.
- Pre-processing filters clean irrelevant or duplicate suggestions.

4.6 Subscription and Notifications

- Users can **subscribe to specific locations** (e.g., “Galata Kulesi”).
- Notifications are sent when new photos are uploaded for that location.

4.7 Mobile Application Features

- Users can **view nearby photos** based on device GPS and camera orientation.
- Swiping navigation between photos is supported.
- Uploads via the mobile app automatically include device metadata (GPS, timestamp).

4.8 Interactive and Multimedia Features

- Users can suggest **textual narrations** describing photos.
- A **chat interface** simulates “talking to the photo,” using contextual data retrieval.
- Photos can be explored in immersive storytelling sequences.

---

5. Non-Functional Requirements

Category	Description
Performance	The system should support concurrent access by at least 1000 users with photo rendering latency < 2s.
Scalability	Cloud-native architecture for growing datasets (>50,000 images).
Security	Role-based access, private photo encryption, HTTPS-only communication.
Usability	Accessible interface with multi-language support and responsive mobile design.
Availability	99.5% uptime for web and API services.
Interoperability	APIs for integration with other archives and cultural databases.

---

6. System Architecture (High-Level Overview)

Core Components:

1. **Frontend (Web + Mobile):** React/Next.js and Flutter/React Native apps.
2. **Backend (API Server):** Django REST Framework + PostgreSQL + ElasticSearch for search.
3. **AI Tagging Module:** Python-based service for image similarity and tagging (TensorFlow / CLIP).
4. **Storage:** Cloud object storage (e.g., AWS S3 / Google Cloud Storage).
5. **Notification Service:** WebSocket or Firebase Cloud Messaging.
6. **Admin Panel:** Moderation tools, suggestion review dashboard.

---

7. Data Model Overview

Entity	Description	Key Attributes
Photo	Represents an uploaded image	`id`, `title`, `location`, `fov_angle`, `date_range`, `tags`, `license`, `visibility`, `source`
Tag	Label for locations or objects	`id`, `name`, `culture_atlas_id`, `visibility`
User	Registered user	`id`, `name`, `role`, `email`, `subscriptions`
Suggestion	User-proposed metadata edits	`id`, `photo_id`, `user_id`, `status`, `content`
Notification	Subscription-based event	`id`, `photo_id`, `user_id`, `timestamp`

---

8. Future Enhancements

- Integration with **OpenStreetMap** and **Google Vision API**.
- Advanced **AI-based date estimation** via scene context learning.
- **Crowdsourced verification badges** for historically validated content.
- Integration with **AR (Augmented Reality)** to overlay historical images on current locations.

---

9. Acceptance Criteria

- Automatic tagging and similarity detection work with ≥85% accuracy.
- Admins can approve or reject suggestions seamlessly.
- Location-based notifications and subscriptions function properly.
- Privacy settings restrict access correctly for private photos.
- Mobile app displays photos relevant to user’s real-world view.
- System scales to 50k+ images with acceptable latency.